Memory Leak (PART II)

AGENDA

I. Detecting Heap Corruption

II. Detecting Memory leaks

III. Custom Leak Detector

IV. Heap Corruption Support

I - Detecting Heap Corruption

Demo:

- Heap corrupt when implement String class
- How to create heap checker class

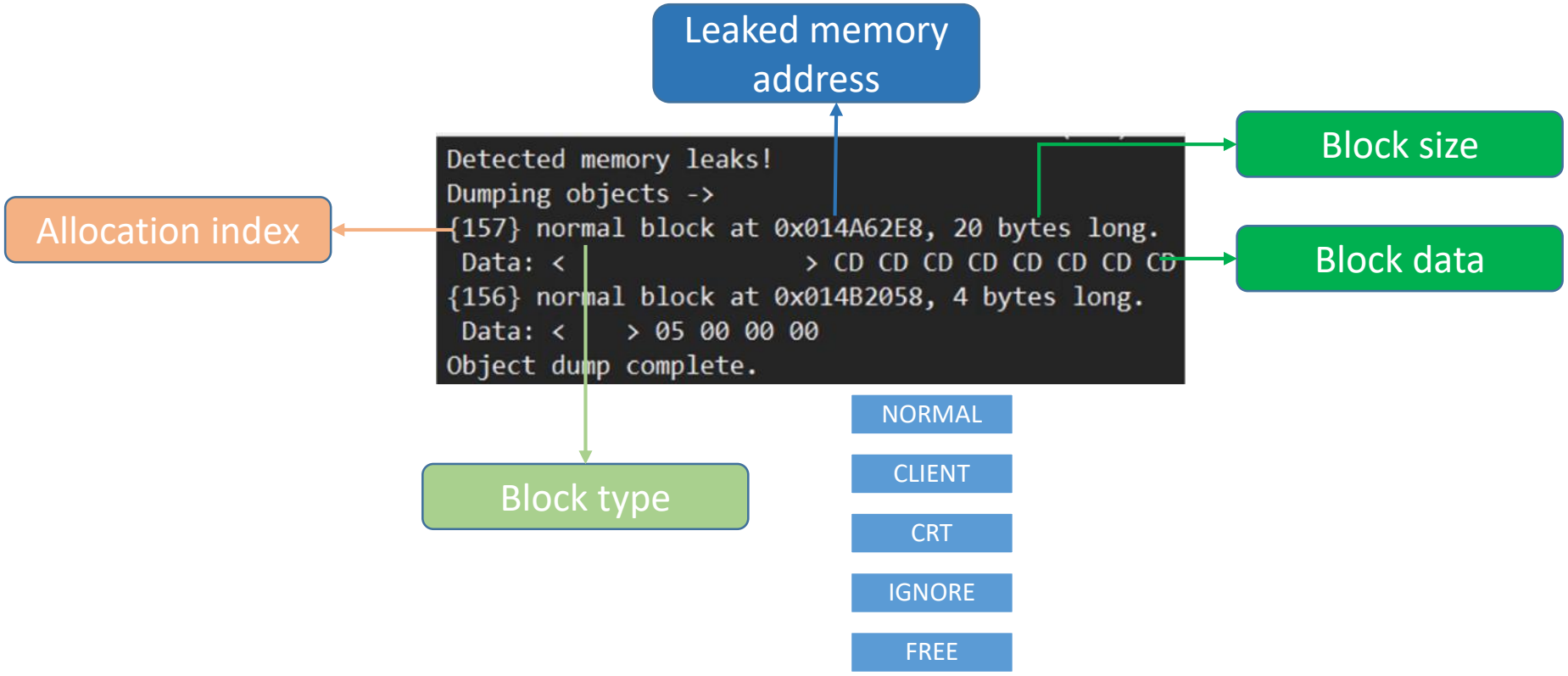
II - Detecting Memory leaks

Detecting Memory leaks

- MinGW provides a function call _CrtDumpMemoryLeaks()
- Just like other CRT debug functions, it works only in debug build
- Once invoked, it goes through all allocated memory blocks
- Any block that is not marked as free is leaked block
  - It dumps diagnostic message in the output windows
  - Return 1

II - Detecting Memory leaks

Detecting Memory leaks (Diagnostic Dump)



II - Detecting Memory leaks

Detecting Memory leaks (Memory Snapshots)

- Debug heap manager provides functions to capture the state of the heap
  - Also called as checkpoint
- Different states can be captured and compared
- If there is a difference between the older & newer heap state, that indicates a possibility of a leak
- Any number of states can be saved & compared; the application can keep on running

II - Detecting Memory leaks

Detecting Memory leaks (Memory Snapshots)

Name1	Description
_CrtMemState	Structure to hold the memory state
_CrtMemCheckpoint()	Captures the current state of the heap in _CrtMemState structure
_CrtMemDifference()	Compares two checkpoints & return 1 if they're different
_CrtMemDumpAllObjectsSince()	Dumps the blocks that have been allocated since an earlier checkpoint
_CrtMemDumpStatistics()	Dumps the difference between block types

II - Detecting Memory leaks

Detecting Memory leaks (CRT Report Mode & type)

Report mode & type

- The heap reporting _Crt functions display different kinds of messages
- Some messages are displayed in IDE output window and some in a separate Window
- The type of messages that are displayed can be controlled by report type
- Report mode can be used to decide where to display the messages

II - Detecting Memory leaks

Detecting Memory leaks (CRT Report Mode & type)

Report type

- There are three report types that can be set by the user
  - _CRT_WARN: represents warnings & other information that do not need immediate attention e.g. memory leaks
  - _CRT_ERROR: unrecoverable problems that require immediate attention e.g. calling abort()
  - _CRT_ASSERT: assertion failures e.g. heap corruption messages

II - Detecting Memory leaks

Detecting Memory leaks (CRT Report Mode & type)

Report Mode

- Following report modes are supported
  - _CRTDBG_MODE_DEBUG: message is written to debugger output window
  - _CRTDBG_MODE_FILE: message is written to a user-defined file
  - _CRTDBG_MODE_WNDW: message is displayed in a message box with **Abort, Retry & Ignore** buttons
  - _CRTDBG_REPORT_MODE: return the current mode for a specific report

III - Custom Leak Detector

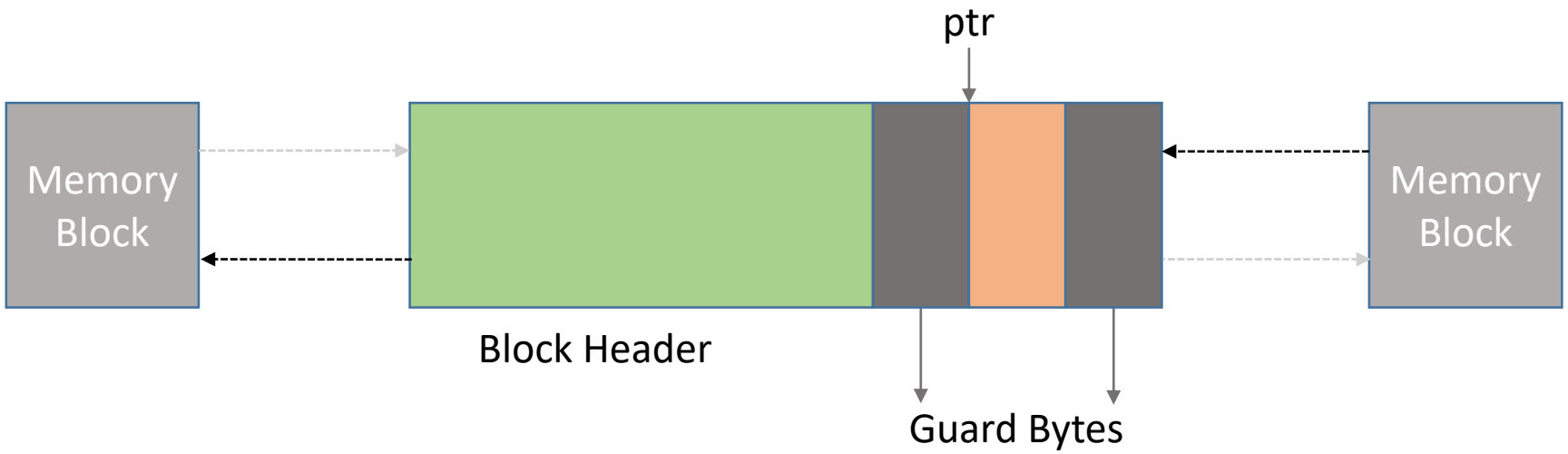
Custom Leak Detector (Implementation)

- Debug heap library detects leaks by storing extra information when memory is requested
- Allocates a bigger memory block that contains the following information
  - Line number
  - File name
  - Type of block
  - Size of requested memory
  - Pointer to previous & next block

III - Custom Leak Detector

Custom Leak Detector (Implementation)

```
int* ptr = new int{6};
```



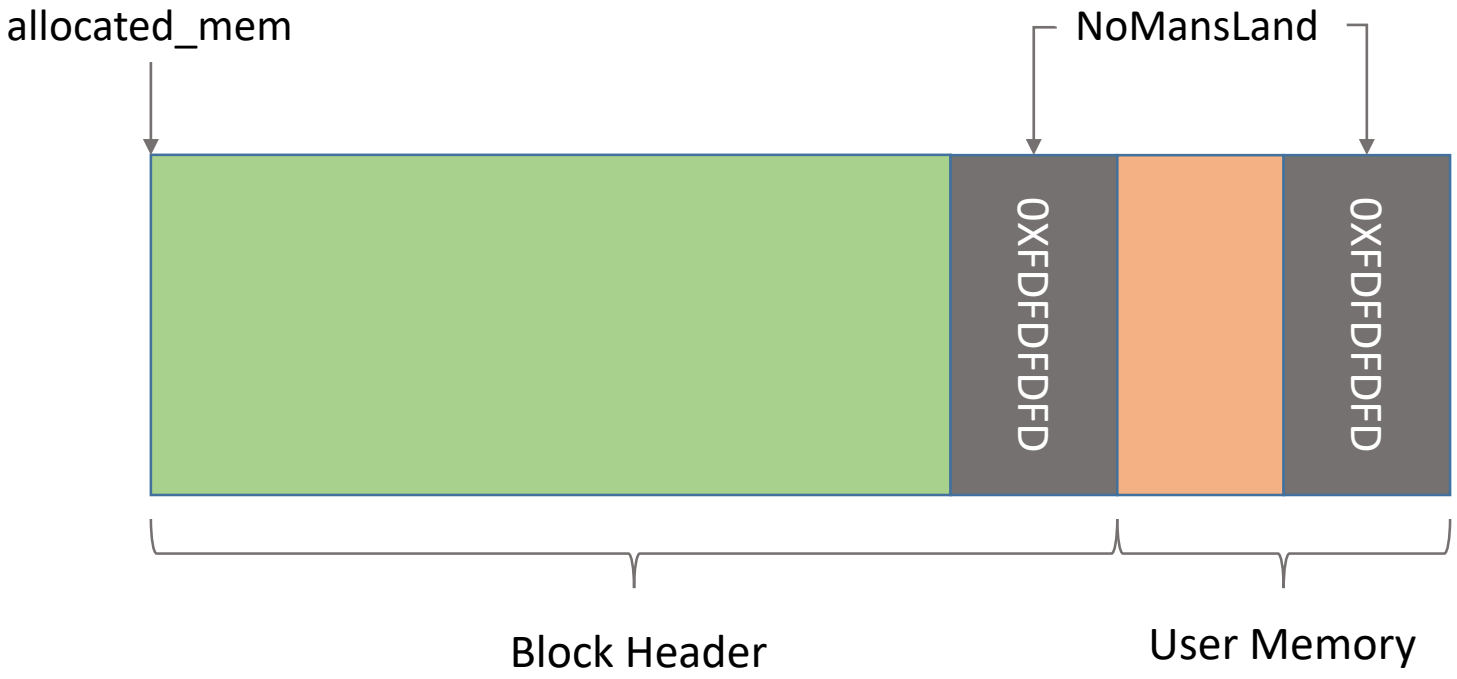
IV - Heap Corruption Support

Heap Corruption Support (Detecting Memory Overwrites)

- Various ways to implement
  - Surround allocated memory with knows values
  - Memory surrounding allocated memory is put in read-only page
- User these approaches to detect memory overwrites
- The user’s allocated memory is surrounded by known values(0xFDFDFDFD)
- This area is called *NoMansLand*
- If the values in *NoMansLand* change, that indicates heap corruption

IV - Heap Corruption Support

Heap Corruption Support (Detecting Memory Overwrites)



THANK YOU